

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

(19) World Intellectual Property Organization  
International Bureau(43) International Publication Date  
5 April 2001 (05.04.2001)

PCT

(10) International Publication Number  
WO 01/24440 A2

(51) International Patent Classification<sup>7</sup>: H04L 12/00 (74) Agent: PILLAY, Kevin; Fasken Martineau DuMoulin LLP, Box 20, Suite 4200, Toronto Dominion Bank Tower, Toronto Dominion Centre, Toronto, Ontario M5K 1N6 (CA).

(21) International Application Number: PCT/CA00/01061

(22) International Filing Date:

18 September 2000 (18.09.2000)

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(25) Filing Language: English  
(26) Publication Language: English  
(30) Priority Data:  
09/409,184 30 September 1999 (30.09.1999) US

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

(71) Applicant (for all designated States except US): MO-SAID TECHNOLOGIES, INC. [CA/CA]; 11 Hines Road, Kanata, Ontario K2K 2X1 (CA).

(72) Inventor; and

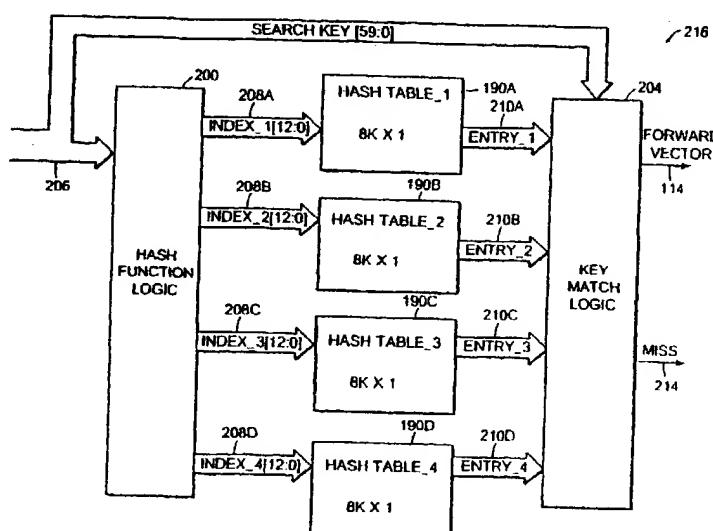
(75) Inventor/Applicant (for US only): BROWN, David, A. [CA/CA]; 110 Allisia Crescent, Carp, Ontario K0A 1L0 (CA).

Published:

— Without international search report and to be republished upon receipt of that report.

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR A FOUR-WAY HASH TABLE



WO 01/24440 A2

(57) Abstract: A number of hash tables are accessed concurrently with a different computed index for each hash table. Each index identifies a location in one of the hash tables at which a forwarding entry may be stored. To search for a forwarding entry, the locations identified by the indices computed by performing a hash function on a search key are searched to a match for the search key. To insert a forwarding entry if single cycle inserting is not required and all the locations identified by the indices computed by performing a hash function on an insert key are used then the data base is reordered so that the forwarding entry can be inserted at one of the locations identified by one of the computed indices for the inset key. If single cycle insertion is required and all the locations identified by the computed indices are used, one of the locations identified by one of the computed indices for the insert key is randomly overwritten by the forwarding entry.



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

## METHOD AND APPARATUS FOR A FOUR-WAY HASH TABLE

## BACKGROUND OF THE INVENTION

In a computer network, a networking switch receives data packets from a number of ingress ports connected to the switch and forwards the data packets to a number of egress ports connected to the switch. The switch determines the egress port to which the data packets are forwarded dependent on a destination address included in the data packet received at one of the ingress ports.

The egress ports to which data packets are to be forwarded based on a destination address are stored in a forwarding entry in a forwarding table in the networking switch. The forwarding table is searched for the forwarding entry associated with a particular destination address.

In general, it is not feasible to provide an entry in the forwarding table for each possible destination address accessible through the switch. For example, for a 48 bit Ethernet destination address, that is, a Media Access Control ("MAC") address,  $2^{48}$  forwarding entries are required in the forwarding table to store forwarding information for all the possible MAC addresses. Instead, only a portion of the MAC addresses are stored in a hash table. A hash function is performed on the destination address to provide a smaller number which is then used to index the hash table.

As a result of performing a hash function on the destination address, multiple destination addresses may map to the same index, that is, aliasing may occur. Thus, an insert of an entry into a location in the hash table may be blocked if the location is already storing a forwarding entry for another destination.

One approach to the problem of aliasing is to generate a second hash function for the destination address in order to provide a second index if the forwarding entry is not found at the first index. However, multiple levels of hashing results in a non-deterministic search by adding delay to the search for the forwarding entry.

One well-known technique for reducing aliasing and providing a deterministic search is to provide more than one location at each index. This is also called the multiple bucket technique. In a hash table implementing multiple buckets, a pre-determined number of locations or buckets are provided at each index.

5 Fig. 1A illustrates a block diagram of a prior art multi-bucket hash table 136. The hash table 136 includes  $2^{15}$  indices (through index bits 14:0) 134. There are four buckets 138A-D at each index 134. Each of the buckets 138A-D stores one forwarding entry. The size of the hash table 136 is determined by the size of the location 138A-D and the total number of indices 134. For example, if each bucket  
10 138A-D is one byte wide, a 128 Kilobytes (32 Kilobytes x 4) hash table is required.

15 However, the multi-bucket hash table results in inefficient use of memory and an insert of a forwarding entry may be blocked if all the locations at an index are used. Thus, no further forwarding entries for destination addresses mapping to the index can be stored at that index in the hash table even though unused locations may be available at other indices in the hash table.

## SUMMARY OF THE INVENTION

20 A network switch includes a plurality of hash tables, each of which is accessed concurrently with a different computed index. The indices are computed by hash function logic from a key dependent on a destination address. A forwarding entry for a key is inserted in one of the locations identified by the computed indices for the key. When a packet is received, the indices are computed to access the four hash tables, and the located forwarding entries are searched for one having a matching key. By appropriate storage of forwarding entries during the insert process, the match search can be completed in one search cycle.

25 The forwarding entry is preferably inserted in an unused location identified by the computed indices by an insert selection logic after detecting the unused location. However, if the insert selection logic determines that all the locations identified by the computed indices for the key are used, the insert selection logic may overwrite one of the used locations.

5

In a preferred embodiment, having determined that all the locations identified by the computed indices for the key are used, the insert selection logic moves one of the forwarding entries having another key to another location available to the other key and then inserts the new forwarding entry. Since the movement of that entry may result in movement of yet another entry, the forwarding entries may be recursively reordered.

10

When a packet is received, a key match logic searches the hash tables for a search key stored in a forwarding entry in one of the locations specified by the computed indices and provides the matching forwarding entry. Because of the reordering in the insert process, the proper forwarding entry in one of the indexed locations can be assured.

The hash function logic preferably computes the index for each hash table by generating a CRC for the key and selecting a portion of the CRC as the index. The number of hash tables and corresponding indices is preferably four.

15

## BRIEF DESCRIPTION OF THE DRAWINGS

20

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1A is a block diagram of a prior art multi-bucket hash table for providing forwarding entries;

25

Fig. 1B is a block diagram illustrating hash tables for providing forwarding entries according to the principles of the present invention;

Fig. 1C is a block diagram of a non-blocking common memory switch in which the hash tables of Fig. 1B can be used;

Fig. 1D is a block diagram of a prior art Ethernet data packet which may be received at an ingress port;

Fig. 2 is a block diagram illustrating the search logic including the hash tables in the forwarding logic shown in the non-blocking common memory switch in FIG. 1C;

5 Fig. 3 is a block diagram illustrating hash function logic shown in FIG. 2;

Fig. 4 is a block diagram illustrating the format of a forwarding entry stored in any one of the hash tables shown in FIG. 1B;

Fig. 5 is a flow graph illustrating the steps in the search logic for searching for a forwarding entry matching a search key in one of the hash tables;

10 Fig. 6 is a block diagram illustrating the insert logic for inserting a new entry in one of the hash tables;

Fig. 7 is a flow chart illustrating the steps in the insert logic shown in FIG. 6 for inserting a forwarding entry into one of the hash tables by overwriting a used entry;

15 Fig. 8 is a flow chart illustrating the steps in the insert logic shown in FIG. 6 for recursively reordering the forwarding entries stored in the hash tables in order to free a location in the hash tables for inserting a forwarding entry.

#### DETAILED DESCRIPTION OF THE INVENTION

Fig. 1B is a block diagram illustrating hash tables 190A-D in a network switch for storing forwarding entries according to the principles of the present invention. Each of the hash tables 190A-D provides only  $2^{13}$  indices, with one location at each index for storing a forwarding entry. The combined hash tables 190A-D have  $12^{15}$  ( $2^{13} \times 2^2$ ) indices. Thus, the four hash tables 190A-D provide the same number of indices as the prior art multi-bucket hash table shown in Fig. 1A, in a hash table that is one fourth the size of prior art multi-bucket hash table. The number of locations at each index and the number of bits per location is a design choice and not a limitation of the invention.

30 Four different indices are computed concurrently by hash function logic dependent on a destination address. The hash function logic is described further with reference to Fig. 3. Each of the computed indices is provided to a different hash table 190A-D. For example, four different indices are generated dependent on Key\_A, that is, Key\_A index 1-4 172A-D. A forwarding entry for a key is inserted

- 5 -

in one of the locations identified by one of the four computed indices so that on a subsequent search for the key, the key is located in one search cycle.

For example, in order to provide a forwarding entry for Key\_A in one search cycle, Key\_A must be stored in one of the locations 138, 146, 156 164 identified by Key\_A Index 1-4 176A-D. If all of the locations 138, 146, 156, 164 are storing forwarding entries for other insert keys, one of the forwarding entries is moved to another location in one of the hash tables 190A-D in order to provide one of the locations 138, 146, 156, 164 for storing Key\_A. A method for reordering the forwarding entries stored in the hash tables 190A-D is described in conjunction with Fig. 8.

As shown, the four Key\_A indices 172A-D identify location 138 in hash table\_1 190A, location 146 in hash table\_2 190B, location 156 in hash table\_3 190C and location 164 in hash table\_4 190D. All the locations 138, 146, 156, 164 are storing forwarding entries for keys other than Key\_A. Key\_B is stored in location 138, Key\_C is stored in location 146, Key\_D is stored in location 156 and Key\_E is stored in location 164. Unused or empty locations are available in the hash tables; for example, locations 140, 142, 144 are unused in hash table 1 190A.

In order to provide a deterministic search, that is, switching without queuing Key\_A must be stored in one of the locations identified by the indices for Key\_A 172A-D. Thus, one of the keys stored in locations 138, 146, 156, 164 must be moved to an unused location, so that a forwarding entry for Key\_A can be stored in the emptied location, and the forwarding entry for the key that is moved is stored in a location identified by one of the moved key's other indices.

Key A shares location 138 with Key B. An entry for Key B stored at location 138 can therefore be moved to an empty location identified by any of the other indices for Key\_B 174B-D. As shown, the other locations 148, 158, 166 identified by the other indices for Key\_B 174B-D are used; therefore, Key\_B can not be moved to one of the other locations 148, 158, 166. However, two of the locations 140, 168 identified by two of the other indices for Key\_C 176A, 176D are empty. Thus, the forwarding entry for Key\_C may be moved from location 146 to location 168 or 140. After moving the forwarding entry for Key\_C, the forwarding entry for Key\_A is inserted into location 146 identified by key\_A index\_2 172B. A

5

search for the forwarding entry for Key\_A or Key\_C is deterministic, that is, each of the searches can be completed in one search cycle. Similarly, forwarding entries for Key\_D could have been moved from location 156 to location 142, 150 or 170 to provide location 156 for inserting Key\_A, or, Key\_E could have been moved from location 164 to location 144, 152 or 162 to provide location 164 for inserting Key\_A.

10

Thus, in order to provide a deterministic search for each forwarding entry stored in the hash tables 190A-D, the forwarding entries stored in the hash tables 190A-D are reordered so that a forwarding entry for a destination address is stored at one of the computed indices for the destination address.

15

A comparison of the approach of the present invention to the prior art approach presented in Fig. 1A is facilitated by considering the expansion of the individual index locations of Fig. 1B to include four buckets each, each as illustrated in Fig. 1A. Such a configuration uses the same 32Kx4 memory as Fig. 1A. Each index into the hash table similarly offers four forwarding entries which might be a proper match to the destination address, but there are four indices into the hash table for each key as opposed to the single index of Fig. 1A.

20

With the present invention, there is a greater sharing of each index with other destination addresses. When inserting forwarding entries in the Fig. 1B approach, conflict at any one index is more likely because of the greater sharing, but overall the Fig. 1B approach with multiple indices per key and multiple buckets provides no more initial conflict in the insertion process. Advantageously, the present approach additionally offers the possibility of shifting otherwise blocking forwarding entries to other locations which are not shared with a particular key, thus offering greater flexibility in the insertion process. Any keys which might be in conflict at a particular index will not likely be in conflict at any other index to which a forwarding entry might be moved. This advantage is obtained regardless of whether multiple buckets are used, and the single bucket approach of Fig. 1B accomplishes this with a reduction in the memory area required by a factor of four.

25

Although the insertion process may require additional processing time for the relocation proportional to the level of recursion which is implementation dependent, time required to process an insertion is much less critical than the need to find a

match in a single cycle during the routing process. With the present invention, such a match is substantially more probable. In fact, a match is almost certain within the bounds of total memory size.

5 Fig. 1C is a block diagram of a common memory switch 100 in which the hash tables shown in FIG. 1B may be used. All data received on ingress ports 102 is stored in segment buffer memory 108 before being switched to one or more egress ports 112. The packet storage manager 106 controls write and read access to the segment buffer memory 108.

10 The switch 100 includes an ingress ports engine 104 and an egress ports engine 110. A data packet is received serially at an ingress port 102. The ingress ports engine 104 detects and processes headers in the received data packet and selects a forward vector 114 stored in a forward entry for the data packet in the forwarding logic 128 in the ingress port engine 104. The forward vector 114 is a bit map, with a bit corresponding to each of the plurality of egress ports 112, indicating whether the data packet is to be forwarded to that egress port 112 dependent on the destination address in the header. The forward vector 114 is forwarded to the packet storage manager 106. For example, data packets with destination address X are forwarded through egress port\_1 112A to the network node with destination address 136A, data packets with destination address Y are forwarded through egress port\_N 112N to the network node with destination address Y 136C and data packets with destination address Z are forwarded through egress port\_N 112N to the network node with destination address Z 136B.

15

20

25 The packet storage manager 106 provides access to the segment buffer memory 108. The packet storage manager 106 provides segment buffer memory addresses 122 for read and write operations to the segment buffer 108 and stores in the manager 106 the locations in the segment buffer memory 108 of each data packet stored. The egress engine 110 selects one of the plurality of egress ports 112, through select control signals 120, on which to transmit a data packet and provides the stored data packet to the selected egress port 112.

30 The segment buffer memory 108 is a common memory shared by all ingress ports 102 and egress ports 112. The switch 100 is non-blocking, that is, a data packet arriving at any of the ingress ports 102 is not blocked from being forwarded

to any of the egress ports 112. The switch 100 provides concurrent processing by the ingress ports engine 104 of data packets received at ingress ports 102 and processing of stored data packets by the egress port engine 110 for egress ports 112.

5 Fig. 1D is a block diagram illustrating a prior art Ethernet data packet 120 which may be received at an ingress port 102. The Ethernet data packet includes a header 122, data field 134, and a frame check sequence 132. The header 122 includes a destination address 124, a source address 126, and a length or type field 130. The source address 126 and the destination address 124 are unique 48 bit addresses identifying the physical device at the source and the destination  
10 respectively. The data packet may also include a 12 bit VLAN Identifier ("VID") (not shown). The size of the data packet 120 is dependent on the size of data field 134, which can vary from 46 bytes to 1,500 bytes.

15 Fig. 2 is a block diagram of the search logic 216 in the forwarding logic 128 shown in the ingress ports engine 104 in the non-blocking common memory switch 100 in FIG. 1C according to the principles of the present invention. The search logic 216 includes four hash tables 190A-D. The number of hash tables 190A-D is a design choice and not a limitation of the invention.

20 The search logic 216 also includes a hash function logic 200 and a key match logic 204. The hash function logic 200 concurrently generates four indices 208A-D dependent on a search key 206. The search key 206 is similar to a tag in cache memory. The key match logic 204 determines which of the four locations identified by the indices 208A-D stores a forwarding entry associated with the search key 206. Upon finding a matching forwarding entry, the forward vector 114 portion of the matching forwarding entry is forwarded to the packet storage manager 106 (Fig.  
25 1C).

30 To search for the forwarding entry corresponding to a destination address in the header of a data packet, the ingress ports engine 104 examines the header 122 (Fig. 1D) of the data packet 120 as the data packet is being received from one of the ingress ports 102 (Fig. 1C). A copy of the 48 bit destination address 124 (Fig. 1D) is concatenated with a 12 bit VLAN IDentifier ("VDI") to form a 60-bit search key 206. The concatenation of the destination address and the VDI is a design choice and not a limitation of the invention, thus, the search key may be the destination

address 124 (Fig. 1D). The search key 206 is forwarded to the hash function logic 200. The hash function logic 200 provides four thirteen bit indices 208A-D as a result of performing a hash function on the search key 206.

5 The invention is described for an Ethernet data packet 120 as shown in FIG. 1D. However, the invention is not limited to Ethernet data packets, it may be used to insert and search for forwarding entries for any other type of data packet.

10 Fig. 3 is a block diagram illustrating the hash function logic 200 shown in FIG. 2. The hash function logic 200 includes a Cyclic Redundancy Check ("CRC") generation logic 300 and a hash table index logic 302. The hash function is a CRC function which is performed on the search key 206 in the CRC generation logic 300. CRC generation logic is well known to those skilled in the art. The hash function is not limited to CRC generation, the hash function used may be any hash function used by those skilled in the art. The CRC generation logic 300 generates a 32 bit CRC 304 from the 60 bit search key 206.

15 The 32 bit CRC 304 is forwarded to the hash table index logic 302. The hash table index logic 302 generates four indices 208A-D from the 32 bit CRC 304. Each of the four indices 208A-D has 13 bits and thus provides 8K addresses to address the 8K locations in each of the hash tables 190A-D. The four indices 208A-D are generated by mapping bits of the 32 bit CRC 304 to the four indices 208A-D  
20 as follows: Bits 12:0 of the CRC 304 are mapped to Bits 12:0 of Index\_1 208A; Bits 18:6 of the CRC 304 are mapped to Bits 12:0 of Index\_2 208B; Bits 25:13 of the CRC 304 are mapped to Bits 12:0 of Index\_3 208C and Bits 31:19 of the CRC 304 are mapped to Bits 12:0 of Index\_4 208D.

25 Returning to Fig. 2, after the indices 208A-D have been generated by the hash function logic 200, the entries 210A-D stored at the locations in the hash tables 190A-D specified by the indices 208A-D are forwarded to the key match logic 204. The search key 206 is also forwarded to the key match logic 204.

30 Fig. 4 is a block diagram showing the format of a forwarding entry 210 stored in any one of the locations in hash tables 190A-D shown in FIG. 2. The forwarding entry 210 includes fields indicating where a data packet is to be forwarded. The forwarding entry 210 includes the following fields: an age flag 402, a remote cache refresh flag 404, a learn port number 406, address associated QoS

valid 408, an address associated QoS 410, a logical port forward vector 412, a static entry indicator flag 414, a type flag 416, a valid entry indicator flag 418 and a key 420.

5 The one bit age flag 402 indicates the age of the forwarding entry 210. The one bit remote cache refresh flag 404 indicates if timeouts are enabled. The learn port number 406 indicates on which interface the MAC address stored in the key 420 in the forwarding entry was learned. For example, if the MAC address was learned on a local external access port, this is the physical ingress port number 104. The address associated Quality of Service ("QoS") field 410 indicates a management assigned quality of service for the address. The one bit address QoS associated valid 10 408 flag indicates whether the QoS field 410 is to be used for future queuing consideration. The 32 bit logical port forward vector 412 indicates to which ports the data packet is to be forwarded. The one bit static entry indicator flag 414 indicates whether the forwarding entry 210 is static or dynamic. The one bit type flag 416 indicates whether the forwarding entry 210 can be used for a level 2 15 forwarding entry. The one bit valid entry indicator flag 418 indicates whether the forwarding entry 210 is valid. The 60 bit key 420 stores the search key 206 associated with the forwarding entry 210.

20 Returning to FIG. 2, the key match logic 204 compares the key entry 420 stored in each of the forwarding entries 210A-D forwarded from the hash tables 190A-D with the search key 206. If there is a match, the 32 bit logical port forward 25 vector 412 (Fig. 4) from that forwarding entry is forwarded on the forward vector 114 to the packet storage manager 106 (Fig. 1). If there is no match a miss signal 214 is generated and the miss is processed by the no match found logic (not shown) in the ingress ports engine 104.

Fig. 5 is a flow graph illustrating the steps in the search logic 216 for searching for a forwarding entry 210 matching a search key 206 in one of the hash tables 190A-D.

30 At step 500, the search logic 216 waits for a search request. If there is a search request processing continues with step 502. If not, the search logic 216 continues to wait for a search request.

At step 502, the hash function logic 200 generates four indices 208A-D from the search key 206 and forwards the four indices 208A-D to the hash tables 190A-D. Processing continues with step 504.

5 At step 504, the forwarding entries 210A-D stored at the locations in the hash tables 210A-D specified by the indices 208A-D are forwarded to the key match logic 204. Processing continues with step 506.

10 At step 506, the key match logic 204 compares the key 420 (Fig. 4) stored in each of the forwarding entries 210A-D with the search key 206 and determines if there is a match. If there is a match processing continues with step 508. If not, processing continues with step 510.

15 At step 508, the forwarding entry 210A-D for the search key 206 has been found. The logical port forward vector 412 (Fig. 4) stored in the forwarding entry 210 is forwarded on the forward vector 114 (Fig. 1C) to the packet storage manager 106 (Fig. 1C). The data packet is forwarded to the predetermined egress ports 112 through the packet storage manager 106, segment buffer memory 108 and the egress ports engine 110. Processing continues with step 500.

20 At step 510, a forwarding entry 210A-D matching the search key 206 has not been found. The ingress ports engine 104 processes the miss for example, by flooding, that is, sending the data packet to all of the egress ports 112. This process is part of the bridging protocol. Processing continues with step 500.

25 Fig. 6 is a block diagram illustrating the insert logic 608 in the forwarding logic 128 (Fig. 1C) for inserting a forwarding entry 210 in a location in one of the hash tables 190A-D. A copy of the 48 bit destination address 124 (Fig. 1D) is concatenated with a 12 bit VLAN IDentifier ("VID") to form a 60-bit insert key 606. The concatenation of the destination address and the VID is a design choice and not a limitation of the invention. The same hash function is performed on the insert key 606 as is performed on the search key 206. The hash function has already been described for the search key 206 in conjunction with Fig. 3. Four indices 208A-D are generated as a result of the hash function. Each of the indices identifies the location in a respective hash table 190A-D at which the insert key 606 may be inserted.

5

The contents of the forwarding entries 210A-D stored at the locations in the hash tables 190A-D identified by the indices 208A-D are forwarded to the forward entry insert logic 600. The forward entry insert logic 600 determines at which locations identified by the indices 208A-D to insert the insert key 606 dependent on the contents of the forwarding entries 210A-D and the state of the overwrite signal 602.

10

The forward entry insert logic 600 determines which of the locations identified by indices 208A-D are unused. If an unused location is found, a forwarding entry 210 including the insert key 606 is inserted in the unused location identified by the index 208 to the unused location. If no unused location is found and the overwrite signal 602 indicates that overwrite is enabled, one of the indices 208A-D is randomly selected and a forwarding entry including the insert key 606 is inserted in the location identified by the selected index 208.

15

If all the locations are used and the overwrite signal 602 indicates that overwrite is not enabled, a reordering of forwarding entries stored in the hash tables 190A-D is performed in order to provide an unused location identified by the indices 208A-D to insert a forwarding entry including the insert key 606. A method for reordering forwarding entries 210A-D in the hash tables 190A-D is described in conjunction with Fig. 8.

20

Fig. 7 is a flow chart illustrating the steps in the insert logic 608 shown in FIG. 6 for inserting a forward entry 210 including an insert key 606 in a location in one of the hash tables 190A-D.

25

At step 700, the hash function logic 200 receives a request to insert a forwarding entry in a location in one of the hashing tables 190A-D. Processing continues with step 702.

At step 702, the hash function logic 200 concurrently generates four indices 208A-D in parallel from the insert key 606. The generation of the indices 208A-D has already been described in conjunction with Fig. 3. Processing continues with step 704.

At step 704, the contents of the locations in the hash tables 190A-D identified by the generated indices 208A-D are forwarded to the forward entry insert logic 600. Processing continues with step 706.

5 At step 716, if the insert key 606 is already stored in a location identified by the generated indices 208A-D a forwarding entry insert is overwritten at that location.

10 At step 706, the forward entry insert logic 600 (Fig. 6) determines from the forwarded contents whether the locations are unused. For example, by examining the valid entry field 418 (Fig. 4) in the forwarding entry 210. If one of the locations is unused, processing continues with step 708. If all the locations identified by the generated indices 208A-D are used, processing continues with step 710.

15 At step 708, the forward entry insert logic 600 (Fig. 6) inserts a forwarding entry 210 including the insert key 606 in a location identified by one of the generated indices 190A-D identifying an unused location.

20 At step 710, the forward entry insert logic 600 (Fig. 6) determines if overwrite is enabled dependent on the state of the overwrite signal 602 (Fig. 6). Overwrite is set if a single cycle insert is required and the forwarding entries in the hash tables are continually updated for example, if the hash tables 190A-D are being used in an Ethernet bridge. If overwrite is enabled, processing continues with step 712. If overwrite is not enabled, processing continues with step 714.

25 At step 712, the forward entry insert logic 600 (Fig. 6) selects an used location identified by any one of the generated indices 208A-D in which to overwrite a forwarding entry including the insert key 606.

30 At step 714, the forward entry insert logic 600 (Fig. 6) performs reordering of the forwarding entries 210A-D stored in locations identified by the generated indices 208A-D in order to move one of the forwarding entries to a location specified by another index in one of the hash tables 190A-D. A method for reordering is described in conjunction with Fig. 8. After the reordering is complete and an unused location is provided at one of the locations identified by the generated indices 208A-D, a forwarding entry 210 including the insert key 606 is inserted in the unused location.

Fig. 8 is a flow chart illustrating the steps in the forward entry insert logic 600 shown in FIG. 6 for reordering the forwarding entries in the hash tables 190A-D shown in Fig. 1B to insert Key\_A in a location identified by the indices for Key\_A 172A-D.

5 At step 800, having determined that Key\_A is not stored at the locations identified by the indices for Key\_A 172A-D, the forward entry insert logic 600 (Fig. 6) determines if any of the locations identified by the indices for Key\_A 172A-D are unused. If there is an unused location, no reordering of the forwarding entries is required and processing continues with step 814. If all the locations are used, 10 reordering is required to provide an unused location identified by one of the indices for Key\_A 172A-D in which to store Key\_A and processing continues with step 802.

15 At step 802, the forward entry insert logic 600 (Fig. 6) determines from Key\_B stored in location 138 identified by Key\_A index\_1 172A and Key\_B index\_1 174A, the other indices for Key\_B 174B-D, in which Key\_B may be stored. The forward entry insert logic 600 may determine the other indices for Key\_B 174B-D by performing a hash function on Key\_B to generate the indices for Key\_B 174B-D or by using Key\_B to index a recursive index table (not shown) in which the indices for Key\_B 174B-D were stored when Key\_B was inserted in location 20 138. Having determined the other indices associated with Key\_B 174B-D processing continues with step 804.

25 At step 804, the forward entry insert logic 600 (Fig. 6) examines the forwarding entries stored at the locations 148, 158, 166 identified by the indices for Key\_B 174B-C. As shown in Fig. 1B, Key\_F is stored at location 148 identified by Key\_B index\_2 174B and Key\_F index\_2 182B, Key\_H is stored at location 158 identified by Key\_B index\_3 174C and Key\_H index\_3 186C and Key\_I is stored at location 166 identified by Key\_B index\_4 174D and Key\_I index\_4 188D. If all the locations 148, 158, 166 identified by the other indices for Key\_B 174B-D are in-use as shown in Fig. 1B, processing continues with step 806. If the forward entry 30 insert logic 600 (Fig. 6) finds one of the locations 148, 158, 166 identified by the other indices for Key\_B 174B-D is unused, processing continues with step 816.

At step 806, the forward entry insert logic 600 (Fig. 6 ) determines from Key\_C stored in location 146, the other indices for Key\_C 806A, 806C-D other than the location identified by the Key\_A index\_2 172B and the Key\_C index\_2 176B, in which Key\_C may be stored. Having determined the other indices associated with Key\_C processing continues with step 808.

5

10

15

20

25

30

At step 808, the forward entry insert logic 600 (Fig. 6) examines the forwarding entries stored at the locations identified by the indices for Key\_C 176A, 176D-C. As shown in Fig. 8, location 140 identified by Key\_C index\_1 176A is unused, Key\_G is stored at location 160 identified by Key\_C Index\_3 176C and Key\_G index\_3 184C and location 168 identified by Key\_C index\_4 176D is unused. If any of the locations 140, 160, 168 identified by the indices for Key\_C 176A, 176C-D are unused as shown in Fig. 8, processing continues with step 810. If the forward entry insert logic 600 (Fig. 6) finds none of the locations 142, 160, 168 identified by the indices for Key\_C 176A, 176C-D are unused, processing continues with step 818.

At step 810, the forward entry insert logic 600 (Fig. 6) moves Key\_C from location 146 identified by Key\_C index\_2 172B and Key\_A index\_2 172B to location 140 identified by Key\_C index\_1 176A. Key\_A is inserted in location 146 identified by Key\_A index\_2 172B.

At step 814, Key\_A is inserted in an unused location identified by one of the Key\_A indices 172A-D.

At step 816, the forward entry insert logic 600 (Fig. 6) moves Key\_B from location 138 identified by Key\_A index\_1 172A and Key\_B index\_1 174A to an unused location identified by one of the indices for Key\_B 174B-D. Key\_A is inserted in location 138 identified by Key\_A index\_1 172A and Key\_B index\_1 174A

At step 818, the forward entry insert logic 600 (Fig. 6 ) determines from Key\_D stored in location 156 identified by Key\_A index\_3 172C and Key\_D index\_3 178D, the other indices for Key\_D 806A-B, 806D, in which Key\_D may be stored. Having determined the other indices for Key\_D processing continues with step 820.

At step 820, the forward entry insert logic 600 (Fig. 6) examines the keys stored at the locations identified by the other indices for Key\_D 806A-B, 806D. As shown in Fig. 1B, all locations 142,150,170 identified by Key\_D indices 806A-B, 806D are unused. If any of the locations 142, 150, 170 identified by the indices for Key\_D 806A-B, 806D are unused as shown in Fig. 8, processing continues with step 822. If the forward entry insert logic 600 (Fig. 6) finds none of the locations 142, 150, 170 identified by the indices for Key\_D 806A-B, 806D is unused, processing continues with step 824.

At step 822, the forward entry insert logic 600 (Fig. 6) moves Key\_D from location 156 identified by Key\_A index\_3 172C to one of the unused locations 142,150,170 identified by Key\_D indices 806A-B, 806D. Key\_A is inserted in location 156 identified by Key\_D index 3 178C.

At step 824, the forward entry insert logic 600 (Fig. 6) determines from Key\_E stored in location 164 identified by Key\_A index\_4 172D and Key\_E index\_4 180D, the other indices for Key\_E 180A-C, 180D, in which Key\_E may be stored. Having determined the other indices associated with Key\_E processing continues with step 826.

At step 826, the forward entry insert logic 600 (Fig. 6) examines the keys stored at the locations identified by the other indices for Key\_E 180A-C. As shown in Fig. 1B, locations 144, 152,162 identified by indices for Key\_E 180A-C are unused. If any of the locations 144, 152, 162 identified by the indices for Key\_E 180A-C are unused as shown in Fig. 1B, processing continues with step 828. If the forward entry insert logic 600 (Fig. 6) finds none of the locations 144, 152, 168 identified by the indices for Key\_E 180A-C is unused, processing continues with step 830.

At step 828, the forward entry insert logic 600 (Fig. 6) moves Key\_E from location 164 identified by Key\_A index\_4 172D and Key\_E index\_4 180D to one of the locations 144, 152, 162 identified by the other indices for Key\_E 180A-C. Preferably, Key\_E is inserted in first empty location detected, that is, location 144 if the indices for Key\_E 180A-C are being searched in the order A-C. Key\_A is inserted in location 164 identified by Key\_A index\_4 172D.

5

At step 830, the index selection logic 600 (Fig. 6) continues to search for an unused location by examining the forwarding entries stored at other locations shared by the Key\_B, Key\_C, Key\_D and Key\_E forwarding entries by examining all the forwarding entries associated with Key\_F, Key\_G, Key\_H and Key\_I until a forwarding entry at one of the Key\_A indices 172A-D is emptied and Key\_A is inserted.

10

In general, the locations identified by the indices for a key are searched in a predetermined order for example, in A-D order. The key is inserted in the first empty location found. Thus, the entry is inserted at the location identified by index\_1 208A, followed by the location identified by index\_2 208B, index\_3 208C and index\_4 208D.

15

Reordering the forwarding entries in the hash tables provides efficient utilization of the hash tables and increases the probability that a forwarding entry can be inserted in the hash tables when the hash tables are almost full.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

## CLAIMS

What is claimed is:

1. A network switch comprising:
  - 5 a plurality of hash tables, each hash table including a plurality of locations, each location capable of storing a forwarding entry accessible by an index; and
  - hash function logic which concurrently computes separate indices for the hash tables from a key dependent on a destination address.
- 10 2. A network switch as claimed in Claim 1 further comprising forward entry insert logic which, upon detecting all locations in the hash tables identified by a computed index are used, overwrites one of the used locations with the forwarding entry.
- 15 3. A network switch as claimed in Claim 2 wherein the location to be overwritten is randomly selected.
4. A network switch as claimed in Claim 1 further comprising forward entry insert logic which, upon detecting that all locations in the hash tables identified by the computed indices are used, moves one of the forwarding entries having another key to another location available to the other key.
- 20 5. A network switch as claimed in Claim 1 further comprising forward entry insert logic which, upon detecting all locations in the hash tables identified by the computed indices are used the insert logic recursively relocates forwarding entries stored in the hash tables so as to provide a location identified by one of the computed indices in which to store the forwarding entry.
- 25 6. A network switch as claimed in Claim 1 further comprising:
  - key match logic which searches locations identified by the computed indices for the key stored in one of the forwarding entries stored in the

locations and upon finding the key provides a forward vector stored at the location in which the key is stored.

7. A network switch as claimed in Claim 1 wherein the hash function logic computes an index by generating a CRC for the key and selecting a portion of the CRC as the index.

8. A network switch as claimed in Claim 1 wherein the number of hash tables is four.

9. A network switch comprising:  
10 a plurality of hash tables, each hash table including a plurality of locations, each location capable of storing one forwarding entry accessible by an index; and

means for concurrently computing separate indices for the hash tables from a key dependent on a destination address.

10. A network switch as claimed in Claim 9 further comprising:  
15 means for inserting a forward entry by overwriting one of the used locations with the forwarding entry, upon detecting all locations in the hash tables identified by a computed index are used.

11. A network switch as claimed in Claim 10 wherein the location to be overwritten is randomly selected.

20 12. A network switch as claimed in Claim 9 further comprising:  
means for inserting a forward entry upon detecting all locations in the hash tables identified by the computed indices are used, by moving one of the forwarding entries having another key to another location available to the other key.

13. A network switch as claimed in Claim 9 further comprising:

means for inserting upon detecting all locations in the hash tables identified by the computed indices are used by recursively relocating forwarding entries stored in the hash tables to provide a location identified by one of the computed indices in which to store the forwarding entry.

5

14. A network switch as claimed in Claim 9 further comprising:

means for searching locations identified by the computed indices for the key stored in one of the forwarding entries stored in the locations and upon finding the key providing the index identifying the location in which the key is stored.

10

15. A network switch as claimed in Claim 9 wherein the means for concurrently computing computes an index by generating a CRC for the key and selecting a portion of the CRC as the index.

16. A network switch as claimed in Claim 9 wherein the number of hash tables is four.

15

17. A method for storing forwarding entries in a network switch comprising:

providing a plurality of hash tables, each hash table including a plurality of locations, each location capable of storing a forwarding entry accessible by an index; and

20

concurrently computing separate indices for the hash tables dependent on a destination address.

18. The method as claimed in Claim 17 further comprising:

inserting a forwarding entry in one of the locations identified by a computed index by overwriting one of the used locations with the forwarding entry upon detecting all the locations are used.

25

19. The method as claimed in Claim 18 wherein the location to be overwritten is randomly selected.

20. The method as claimed in Claim 17 further comprising:

inserting a forwarding entry in one of the locations identified by a computed index by moving one of the forwarding entries having another key to another location available to the other key upon detecting all the locations are used.

5

21. The method as claimed in Claim 17 further comprising:

inserting a forwarding entry in one of the locations identified by a computed index by recursively relocating forwarding entries stored in the hash tables so as to provide a location identified by one of the computed indices in which to store the forwarding entry, upon detecting all the locations identified by the computed index are used.

10

22. The method as claimed in Claim 17 further comprising:

searching locations identified by the computed indices for the key stored in one of the forwarding entries stored in one of the locations; and

15

providing the index identifying the location in which the key is stored, upon finding the key.

23. The method as claimed in Claim 17 wherein computing further comprises:

generating a CRC for the key; and

selecting a portion of the CRC as the index.

20

24. The method as claimed in Claim 17 wherein the number of hash tables is four.

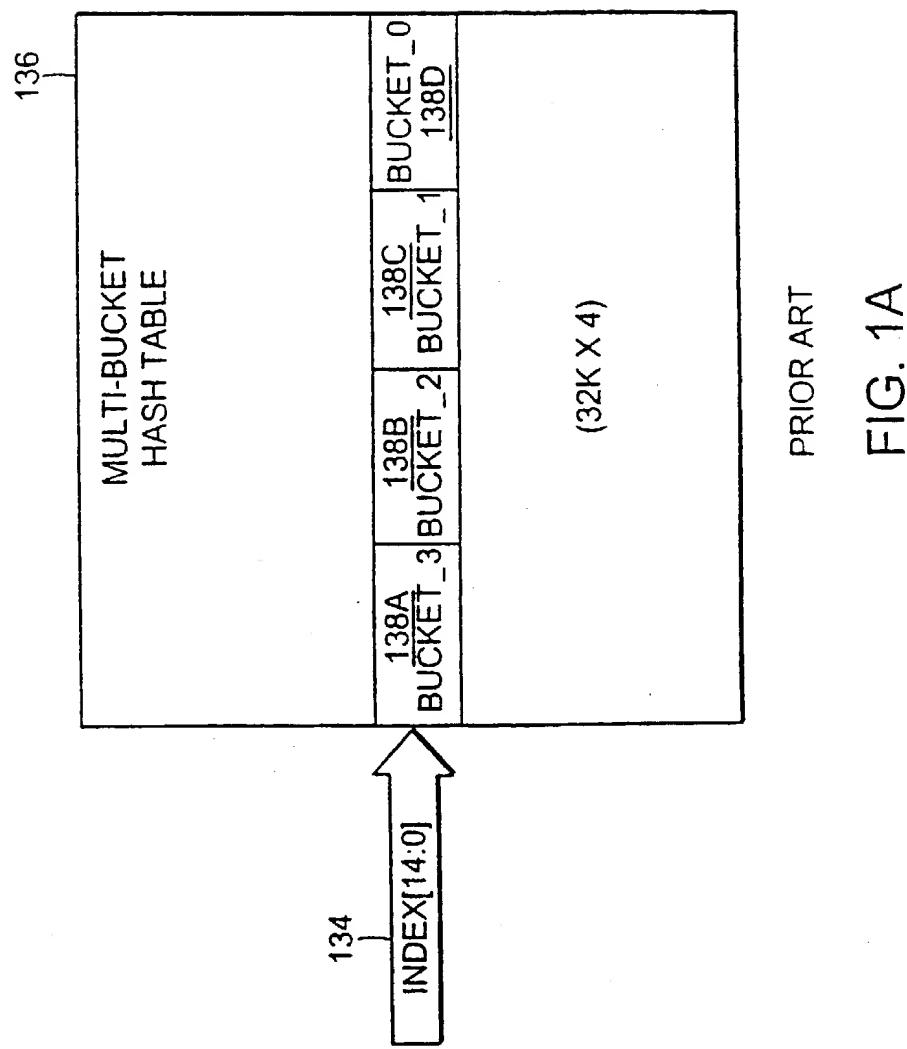


FIG. 1A

PRIOR ART

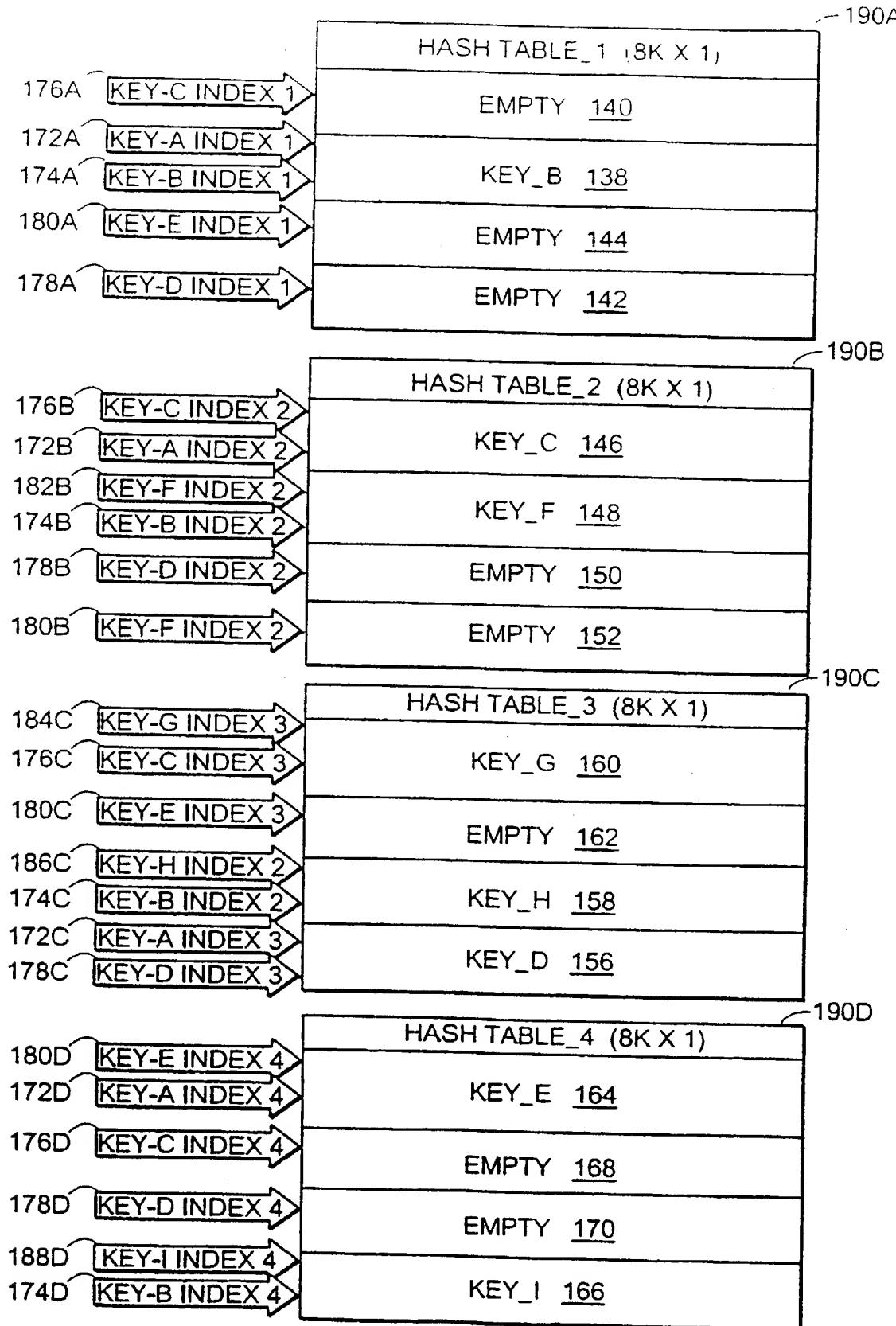


FIG. 1B

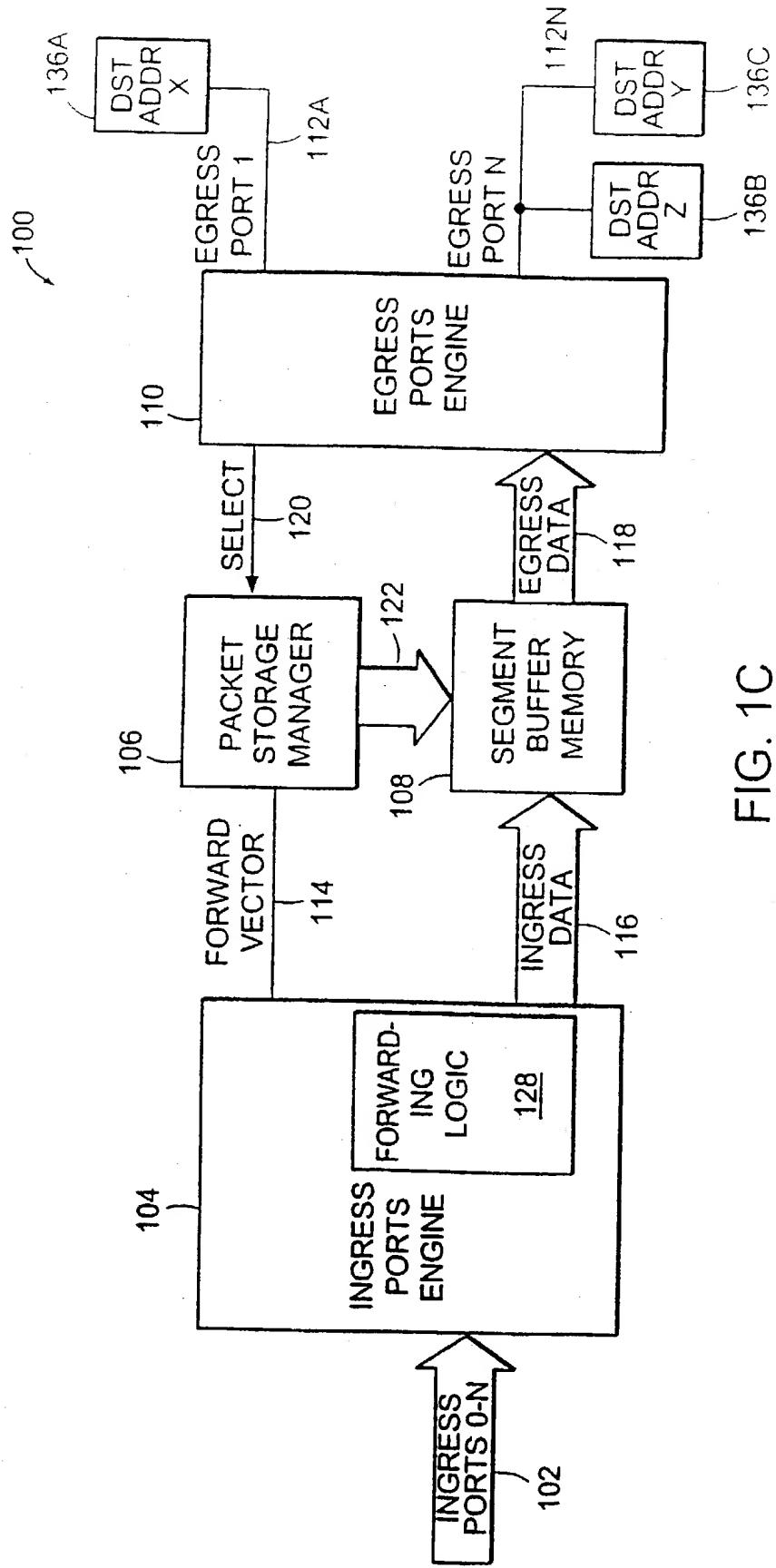
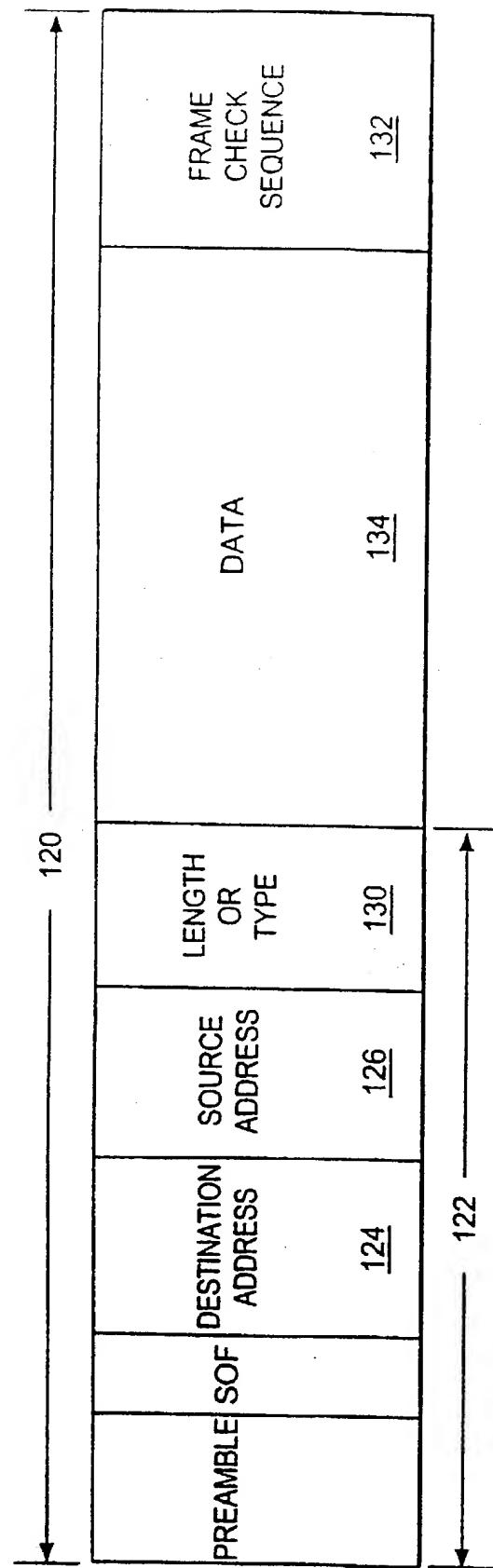


FIG. 1C

PRIOR ART  
FIG. 1D

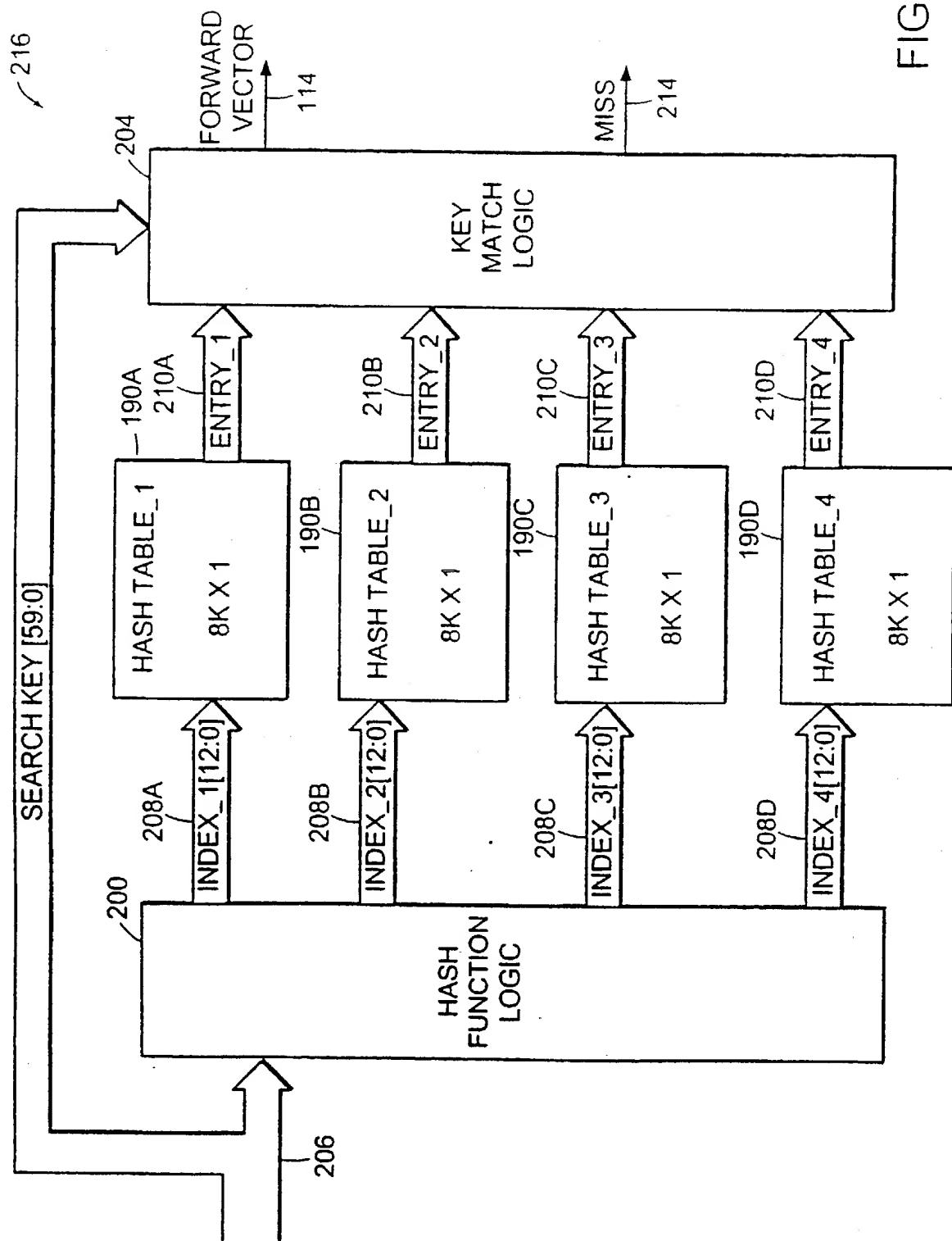


FIG. 2

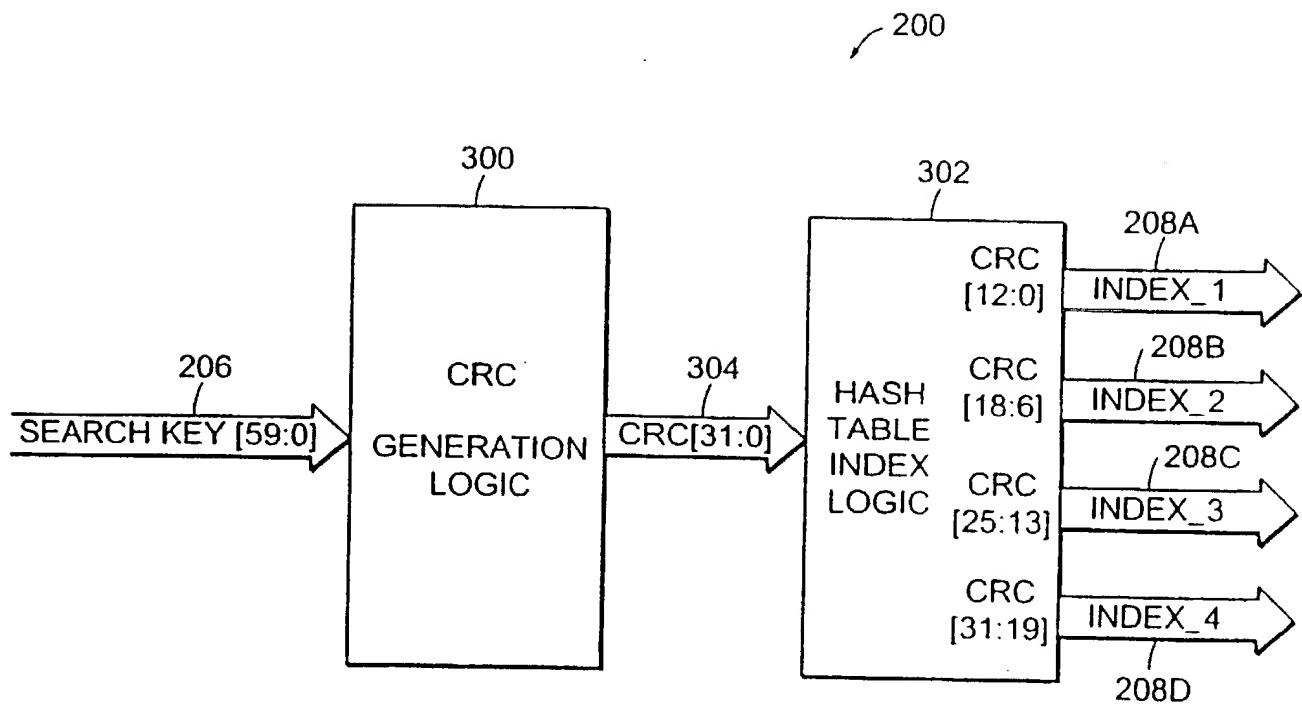


FIG. 3

105	RE-MOTE AGE FLAG	CACHE RE-FRESH FLAG	LEARN PORT NUMBER	ADDR. ASSOC QOS	ADDR. ASSOC QOS	LOGICAL PORT FORWARD VECTOR	NOT USED ENTRY	STATIC ENTRY	TYPE	VALID ENTRY	KEY
210	402	404	406	408	410	412	414	416	418	420	

FIG.4

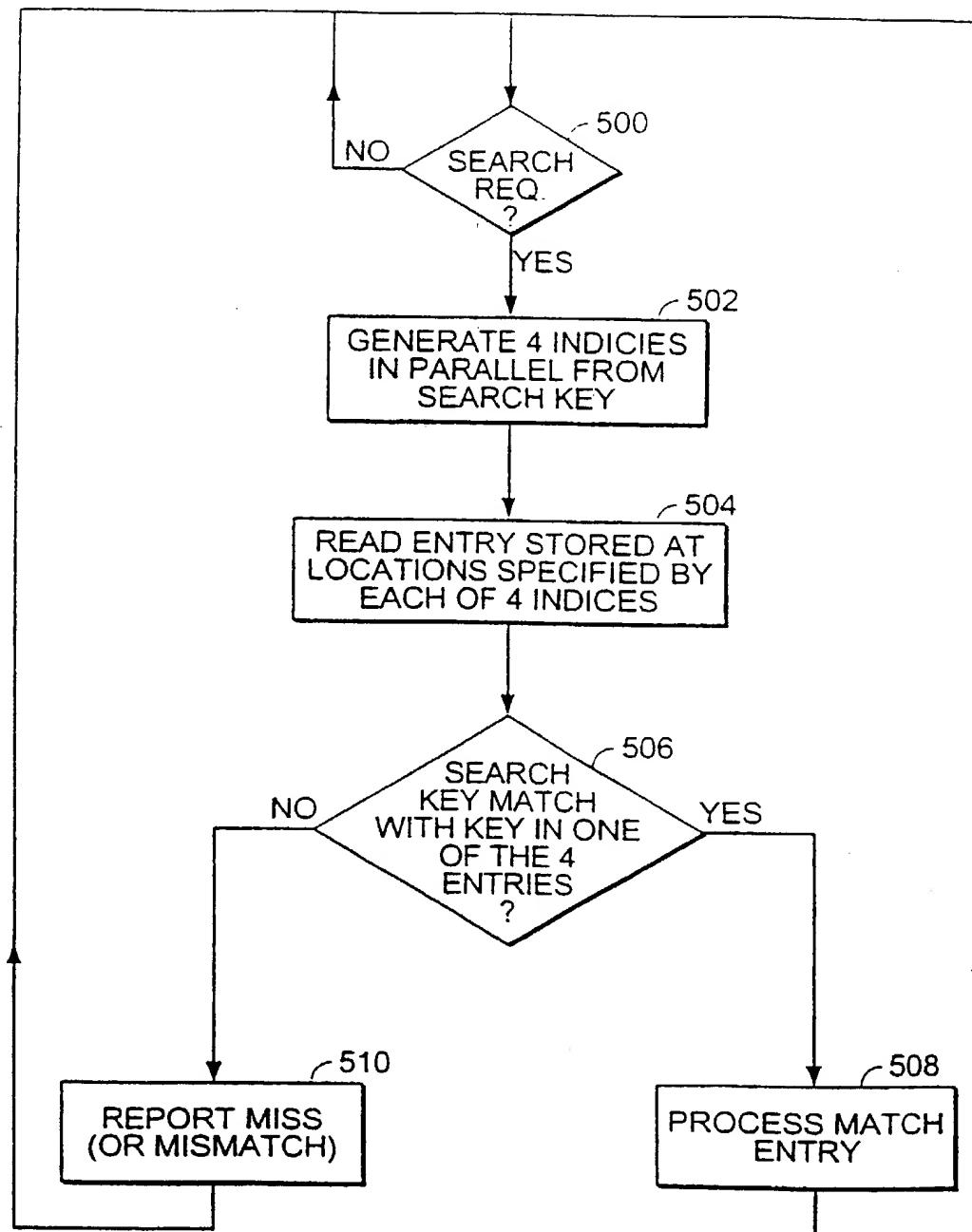


FIG. 5

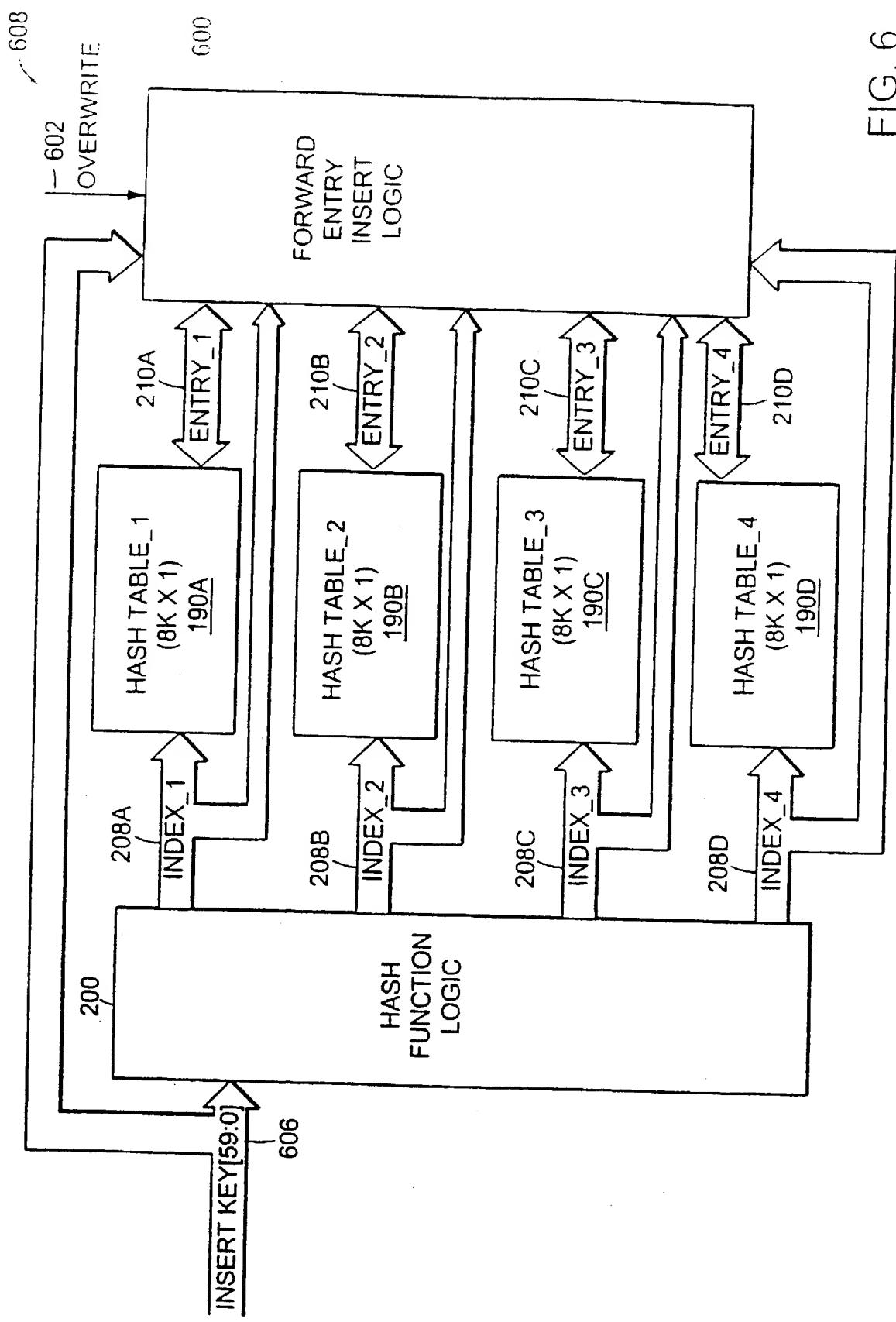


FIG. 6

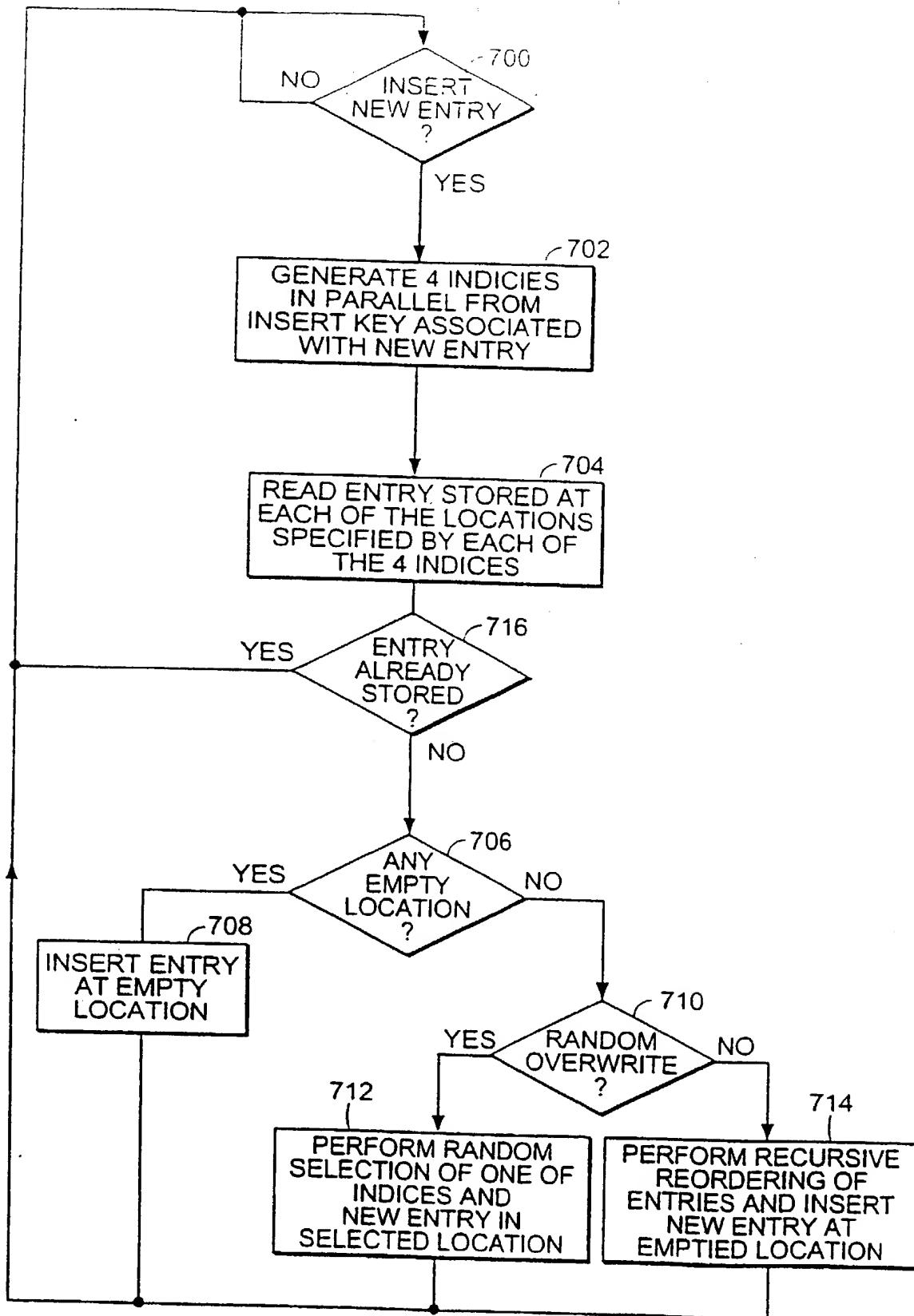


FIG. 7

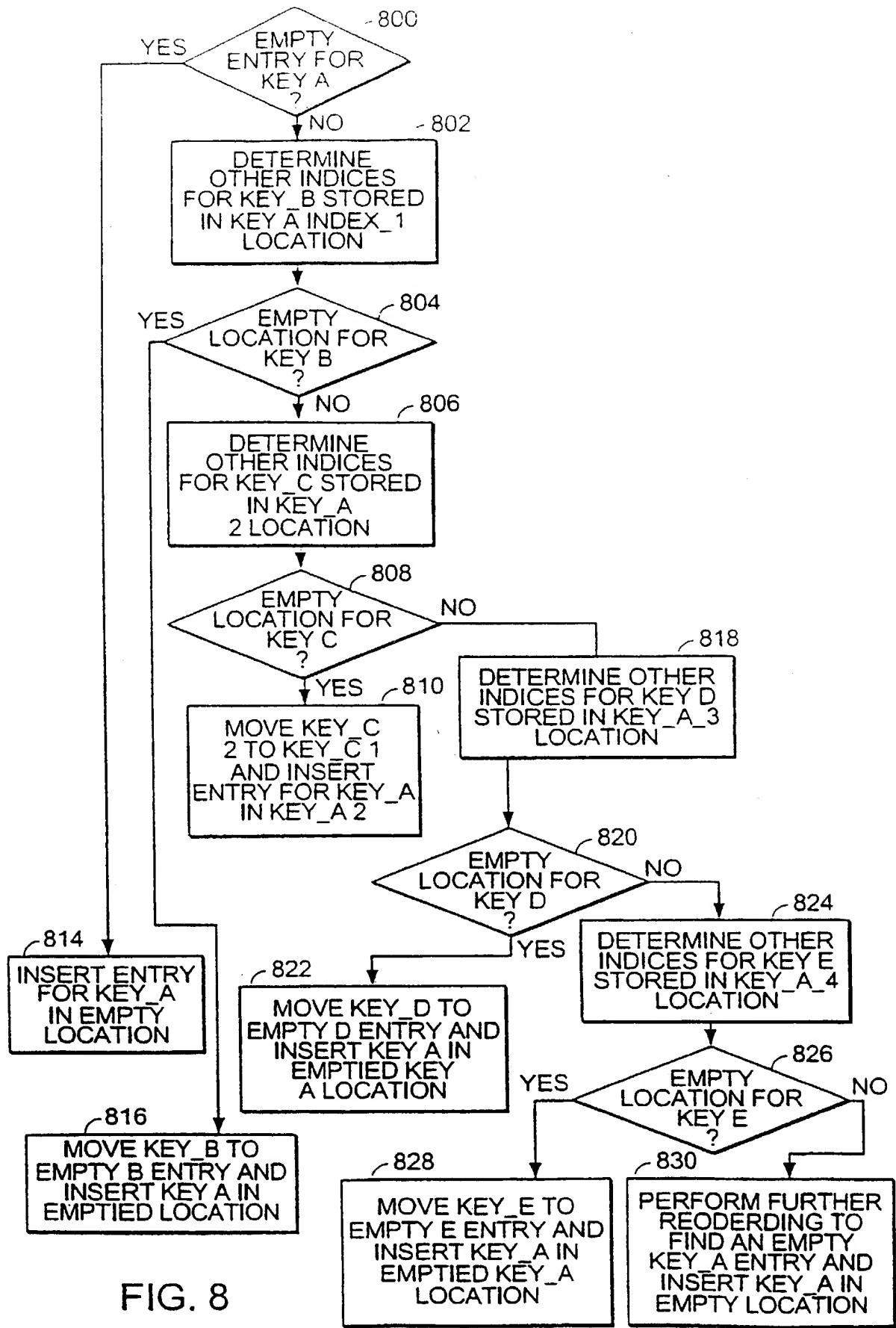


FIG. 8

(19) World Intellectual Property Organization  
International Bureau(43) International Publication Date  
5 April 2001 (05.04.2001)

PCT

(10) International Publication Number  
WO 01/24440 A3(51) International Patent Classification<sup>7</sup>: H04L 12/56, G06F 12/10, 17/30, H04L 29/06, G06F 12/08

(74) Agent: PILLAY, Kevin: Fasken Martineau DuMoulin LLP, Box 20, Suite 4200, Toronto Dominion Bank Tower, Toronto Dominion Centre, Toronto, Ontario M5K 1N6 (CA).

(21) International Application Number: PCT/CA00/01061

(22) International Filing Date:  
18 September 2000 (18.09.2000)

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
09/409,184 30 September 1999 (30.09.1999) US

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

(71) Applicant (for all designated States except US): MO-SAID TECHNOLOGIES, INC. [CA/CA]; 11 Hines Road, Kanata, Ontario K2K 2X1 (CA).

## Published:

— with international search report

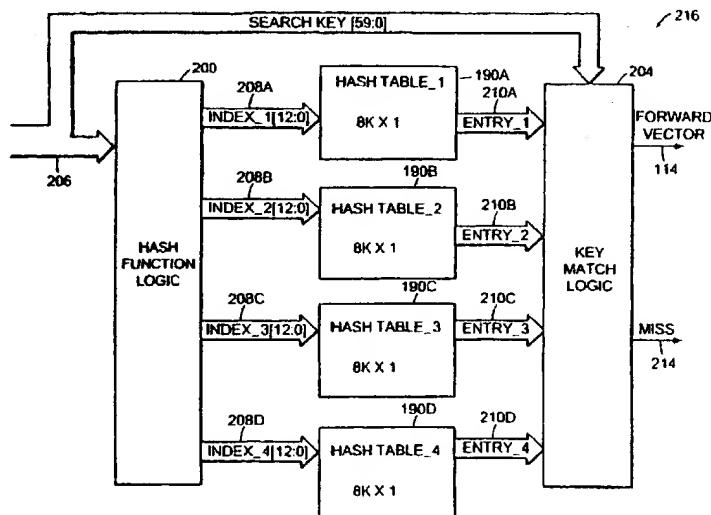
(72) Inventor; and

(75) Inventor/Applicant (for US only): BROWN, David, A. [CA/CA]; 110 Allisia Crescent, Carp, Ontario K0A 1L0 (CA).

(88) Date of publication of the international search report:  
8 November 2001

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR A FOUR-WAY HASH TABLE



(57) Abstract: A number of hash tables are accessed concurrently with a different computed index for each hash table. Each index identifies a location in one of the hash tables at which a forwarding entry may be stored. To search for a forwarding entry, the locations identified by the indices computed by performing a hash function on a search key are searched to a match for the search key. To insert a forwarding entry if single cycle inserting is not required and all the locations identified by the indices computed by performing a hash function on an insert key are used then the data base is reordered so that the forwarding entry can be inserted at one of the locations identified by one of the computed indices for the insert key. If single cycle insertion is required and all the locations identified by the computed indices are used, one of the locations identified by one of the computed indices for the insert key is randomly overwritten by the forwarding entry.

WO 01/24440 A3



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

A. CLASSIFICATION OF SUBJECT MATTER					
IPC 7	H04L12/56	G06F12/10	G06F17/30	H04L29/06	G06F12/08

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category <sup>o</sup>	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 032 987 A (BRODER ANDREI Z ET AL) 16 July 1991 (1991-07-16)  column 1, line 53 - line 68 column 3, line 35 - line 61 figure 4	1,4,6,8, 9,12,14, 16,17, 20,22,24
Y		2,3,7, 10,11, 15,18, 19,23
A	---	5,13,21
Y	US 5 802 054 A (BELLENGER DONALD M) 1 September 1998 (1998-09-01) Column 6, line 67 -column 7, line 8 ---	7,15,23
		-/-

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

## ° Special categories of cited documents :

- °A° document defining the general state of the art which is not considered to be of particular relevance
- °E° earlier document but published on or after the international filing date
- °L° document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- °O° document referring to an oral disclosure, use, exhibition or other means
- °P° document published prior to the international filing date but later than the priority date claimed

- °T° later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- °X° document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- °Y° document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- °&° document member of the same patent family

Date of the actual completion of the international search

1 June 2001

Date of mailing of the international search report

15/06/2001

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel: (+31-70) 340-2040, Tx. 31 651 epo nl  
Fax: (+31-70) 340-3016

Authorized officer

Siebel, C

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP 0 365 337 A (HEWLETT PACKARD CO) 25 April 1990 (1990-04-25) claim 4 -----	2,3,10, 11,18,19

• Patent document cited in search report	Publication date	Patent family member(s)			Publication date
US 5032987	A	16-07-1991	NONE		
US 5802054	A	01-09-1998	US	5949786 A	07-09-1999
EP 0365337	A	25-04-1990	DE	68925557 D	14-03-1996
			DE	68925557 T	05-06-1996
			US	5027350 A	25-06-1991